# Zero Install

The no-fuss, stateless, decentralised package manager

Tim Cuthbertson

`gfxmonk.net`

# Why am I here?

- I think existing package managers aren’t up to the job
- I think zero install is awesome. I use it as often as I can
  - But we need more packages, so we need more developers to make packages for it
  - ... hi there!

# Who authors open-source software?

Show of hands...

# ..and who also makes packages for their software?

Show of hands...

# Everyone should!

(probably)

- It helps people run your code
- It helps people *reuse* your code
- It's a critical part of open source software
- *"But it's* **boring**, *and I'm* **lazy***"*
- ..and there are so many options

# Types of Package managers

Two types of package / dependency managers:

- System based
- Language based

# System-based package managers

Think `apt`, `yum`, `fink`, etc... The backbone of linux distributions.

But they all:

- Require root access
- Have a centralised, "trusted" repository
- Have colliding names
  (`git` in fedora is `git-core` in debian due to a clash)
- Have colliding *files*
  - "You have a `/usr/bin/python`? I have one of those too!"
  - damn.

# System-based package managers

Very organised (hopefully)

- Works well enough for distributions
- Centralised, coordinated per release
- But it sucks for end users wanting the latest version of something
  - Build from source
  - Use a PPA

# Language-based package managers

Ruby gems, maven poms, python eggs, etc...

- No root access required!
- But mainly for libraries, not apps
- .. so apps have to repackage them
- Polyglot apps are usually not supported
- New language? New package manager.

Also, these systems often suck. Setuptools won't let you uninstall things. Cabal requires a PhD, etc...

# So What Happens?

- Many developers use their language package-manager of choice (or nothing at all)
  - It's easier
  - You don't need permission to publish
  - We're all pretty lazy
- Distros get the joy of repackaging these into deb, rpm, etc..
- Now you have 5 different `libawesome`$^{\text{TM}}$packages in various formats
- ..and then users whinge about out-of-date packages

# And after all that?

Windows and mac users still don't get any dependency management for their applications.

(but that's okay, they're used to it)

# That Sucks.

I know! Let's make a super package format that can understand .rpm, .deb, pom.xml, and handle dependencies between all of them!

- No, that's a terrible idea.
- And it's been tried. Badly.
- We need something better than the current crop.

# Zero Install to the Rescue?

Yes, it's *Yet Another Package Manager*. But but but:

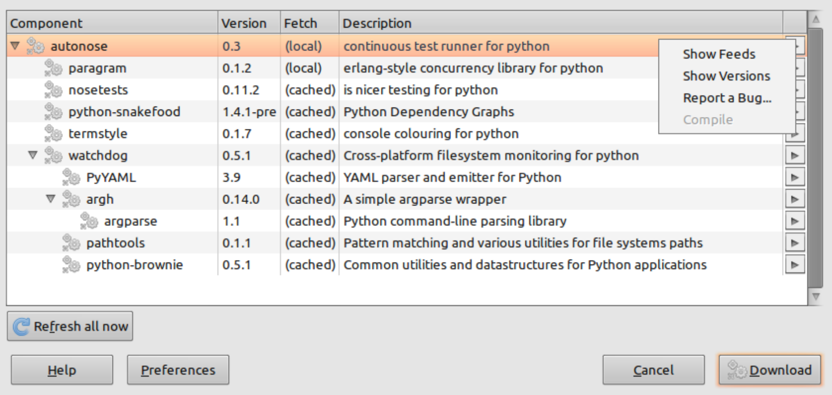
- Cross platform and user-friendly
- Language agnostic
- Supports both source and binary packages
- Any user can run new software (no root required)
- Distributed - a package's id is just its URL, no central repository
- Packages are required to be read-only and have no hard-coded paths
  - This allows for multiple concurrent versions for free
  - No installation required - just cache it and run locally
  - Interesting opportunities for sandboxing (see `http://zero-install.sourceforge.net/ebox.html`)

# Zero Install

- It doesn’t hate your system package manager
  - It can reuse already-installed packages
  - System packages can “implement” zero-install packages
- It’s really decentralised
  - You can depend on any URL
  - You can host a 0install repo on any web server
- ..and it’s user friendly.

# Zero Install

Example zero install “first run” window:



| Component | Version | Fetch | Description |
|---|---|---|---|
| ▽ autonose | 0.3 | (local) | continuous test runner for python |
| paragram | 0.1.2 | (local) | erlang-style concurrency library for python |
| nosetests | 0.11.2 | (cached) | is nicer testing for python |
| python-snakefood | 1.4.1-pre | (cached) | Python Dependency Graphs |
| termstyle | 0.1.7 | (cached) | console colouring for python |
| ▽ watchdog | 0.5.1 | (cached) | Cross-platform filesystem monitoring for python |
| PyYAML | 3.9 | (cached) | YAML parser and emitter for Python |
| ▽ argh | 0.14.0 | (cached) | A simple argparse wrapper |
| argparse | 1.1 | (cached) | Python command-line parsing library |
| pathtools | 0.1.1 | (cached) | Pattern matching and various utilities for file systems paths |
| python-brownie | 0.5.1 | (cached) | Common utilities and datastructures for Python applications |

# Zero Install

And it's easy to develop with:

- It's unobtrusive
  - You don't **install** anything, so there's no global state to break
  - No root privileges means you can use a dummy user as clean sandbox
  - Your source code doesn't have to know about 0install
  - Dependencies are provided by setting environment variables - `$PATH`, `$CLASSPATH`, `$PYTHON_PATH`, `$GEM_PATH`, `$LD_LOAD_PATH`, etc...
  - Local feeds allow you to always use zero install's dependency management, even for development

# Zero Install

Side note: protecting developers (and deployed applications) from global state

- Python virtualenv
- Ruby rvm + rubygems + bundler
- And they're not even very good
  - Don't get me started on rvm...
- A terrible option for client-side apps

# Zero Install

No global state means you can even use it internally

- Instant dependency resolver for your browser / library / media player / IDE plugins...
  - It even handles plugins-that-depend-on-libraries

(Nobody wants to write another plugin system, they just think they have to)

# An admission

Sadly, zero install isn't perfect. If you actually *want* global state, you're on your own. That means zero install can't help you with:

- System configuration

..but that's a pretty short list.

# Zero Install

If you only make one package, make it a good one.

# Zero Install

Comments? Questions?

Tim Cuthbertson
`gfxmonk.net`
`@gfxmonk`

Zero install
`0install.net`